**Documentation: Analysis report**

**Author: Uwe Ricken**

**Version: 1.00**

**Date of documentation: 02.11.2017**

**Date of approval: 07.11.2017**

**Approved by: Uwe Ricken – db-Berater GmbH**

**Copyright: db-Berater GmbH**

Unless otherwise specified, the documents are for your personal and non-commercial use. You may not modify copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell any information, software, products or services.

# Introduction

**db** Berater GmbH[[1]](#footnote-1) is specialized in database technologies of Microsoft. **db** Berater GmbH is working with certified employees and partners for the support of customers of the technical environment of Microsoft SQL Server. We are counselling our customers in the implementation of new SQL Server systems, analysis of existing systems and develop customized database systems based on Microsoft SQL Server.

# The author of this report

The analysis report has been written by Uwe Ricken, Managing Director of **db** Berater GmbH. Uwe is working with Microsoft SQL Server since 1998 and has a fundamental knowledge of its technique for more than 15 years. Uwe has achieved the following certificates for Microsoft SQL Server

* Microsoft Certified Master: CHARTA ‑ Data Platform
* Microsoft Certified Master: SQL Server 2008
* Microsoft Certified Solutions Expert: Data Platform
* [Most Valued Professional: Data Platform](https://mvp.microsoft.com/de-de/PublicProfile/5000190)

Table of content

[Introduction 2](#_Toc497821303)

[The author of this report 2](#_Toc497821304)

[Order Description 4](#_Toc497821305)

[Result of system analysis (abstract) 4](#_Toc497821306)

[Technical configuration 4](#_Toc497821307)

[Hardware 4](#_Toc497821308)

[Virtualisation 4](#_Toc497821309)

[Microsoft Windows Operating System 4](#_Toc497821310)

[Microsoft SQL Server 5](#_Toc497821311)

[Survey result 5](#_Toc497821312)

[Operating System 5](#_Toc497821313)

[Servicepack 5](#_Toc497821314)

[HDD / Storage 5](#_Toc497821315)

[Instant File Initialization 5](#_Toc497821316)

[Traceflags 5](#_Toc497821317)

[Virtual Volumes 7](#_Toc497821318)

[Tempdb Size and Placement Recommendations 8](#_Toc497821319)

[Memory / RAM 8](#_Toc497821320)

[NUMA-architecture 9](#_Toc497821321)

[Microsoft SQL Server 10](#_Toc497821322)

[Configuration settings of Microsoft SQL Server 10](#_Toc497821323)

[Cost Threshold for Parallelism 11](#_Toc497821324)

[Max Degree of Parallelism 11](#_Toc497821325)

[Min/Max Server Memory 11](#_Toc497821326)

[xp\_cmdshell 12](#_Toc497821327)

[Wait Stats 12](#_Toc497821328)

[Databases 14](#_Toc497821329)

[TEMPDB 14](#_Toc497821330)

[User database(s) 15](#_Toc497821331)

[Recommendation 16](#_Toc497821332)

[Activation of default Traceflags 16](#_Toc497821333)

[Cost Threshold for Parallelism 17](#_Toc497821334)

[Max Degree of Parallelism 17](#_Toc497821335)

[Deaktivierung des Systemkontos [sa] 17](#_Toc497821336)

[Security 17](#_Toc497821337)

[Deaktivierung von xp\_cmdshell 18](#_Toc497821338)

[TEMPDB – dediziertes Laufwerk für Datenbank/Protokoll 18](#_Toc497821339)

[Zusammenfassung 18](#_Toc497821340)

Uwe Ricken has been instructed by <CUSTOMER\_NAME> to check the configuration and the performance of a Microsoft SQL Server. The analysis started at 11/02/2017 at 1:00PM.

# Order Description

<CUSTOMER\_NAME> is running an application on a virtualized Microsoft SQL Server environment with a dedicated test- and production system.

Information

The SQL Server has been restarted at 2017-10-17 01:22. All results are based on an uptime until 2017-11-01. Predictable long-term influences cannot be valued by the collected performance counters!

# Result of system analysis (abstract)

* Missing Service Pack for Microsoft SQL Server 2012[[2]](#footnote-2)
* Important trace flags for startup are not configured for the SQL Server (page 6)
* High drive latency on DATA storage
* Databases are running in mode of SQL 2005
* Databases are running in mode of SQL 2008
* Wrong configured configuration values for parallelism (page 9)
* High proportion of parallel queries >65%
* Wrong configuration of NUMA

# Technical configuration

## Hardware

The physical hardware was not a component of the analysis.

## Virtualization

The settings/configuration of the host system was no component of the analysis but should be run separately to ensure a configuration under “best practice” for VMware.

## Microsoft Windows Operating System

Microsoft SQL Server 2012 is running with Windows Server 2012. It is configured with 16 vCores and 64 GB RAM.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPU | PhysicalMem\_GB | CommittedMem\_GB | CommittedTargetMem\_GB | max\_workers\_count |
| 16 | 64.00 | 58.54 | 58.59 | 704 |

## Microsoft SQL Server

The researched System is a Microsoft SQL Server 2012 Enterprise Edition (64bit). SQL Server is installed as a DEFAULT instance with the actual patch level 11.0.6518.0.

# Survey result

The research was following a „top-down-method“. Due to lack of permissions the focus was only on the SQL Server and its fundamental environment settings.

## Operating System

Microsoft SQL Server is running on an ESX-Host. A survey of the Hypervisor couldn’t be done because of missing permissions on the box.

## Service pack

Microsoft SQL Server is running with SP3 (CU1) from January 2016!). It is recommended to update to SP4 from October 2017.

## HDD / Storage

Microsoft SQL Server is running with 5 virtual disks (C:, D:, E:, F:, H:). The system databases (master, model, msdb) are hosted on Drive D: TEMPDB is located on drive H: All user databases are hosted on drives E: (data) and F: (logs). The author cannot investigate the formatted block size of the storage! Best practice is a block size of 64Kbytes for the data and log disks

## Instant File Initialization

This option allows for much faster data file allocations (CREATE AND ALTER FILE) but DOES NOT WORK FOR LOG FILE ALLOCATIONS. This is enabled for each instance via the "Perform volume maintenance tasks" local security policy.

This permission prevents SQL Server from "zeroing out" new space when a data file will be created or expanded (**it is not applied to log files**). This helps performance for CREATE DATABASE, ALTER DATABASE, RESTORE, and AUTOGROW. It can have a significant positive impact on how long it takes to create or expand a data file, but there is a small security risk in doing so. That is because a file "delete" just deallocates the space and a new allocation can reuse that space which may still have data in it.

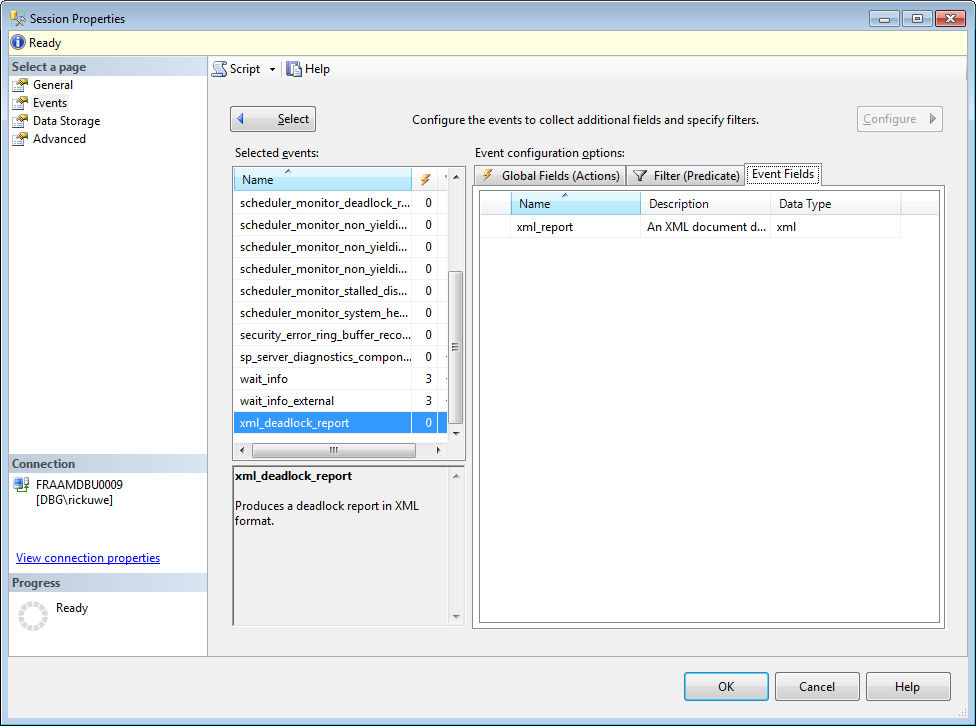
## Trace flags

Trace flags are used to temporarily/permanent set specific server characteristics or to switch off a particular behavior. Trace flags are frequently used to diagnose performance issues or to debug stored procedures or complex computer systems.

The following enabled trace flags have been detected in the UAT environment:

|  |  |
| --- | --- |
| Trace flag | Description |
| 1204 | Returns the resources and types of locks participating in a deadlock and also the current command affected.  **Scope:** global only |
| 1222 | Returns the resources and types of locks that are participating in a deadlock and also the current command affected, in an XML format that does not comply with any XSD schema.  **Scope:** global only |
| 3226 | By default, every successful backup operation adds an entry in the SQL Server error log and in the system event log. If very frequent log backups are created, these success messages accumulate quickly, resulting in huge error logs in which finding other messages is problematic. This trace flag suppress these log entries.  **Scope:** global only |

The above-mentioned trace flags 1204 and 1222 are useful for deadlock detection. Basically, this is not required anymore because deadlocks will be monitored with the generic “system health” extended event session!



Picture 1: Configuration of deadlock monitoring in system health extended event session

The system health session is an Extended Events session that is included by default with SQL Server. This session starts automatically when the SQL Server Database Engine starts, and runs **without any noticeable performance effects**. The session collects system data to troubleshoot performance issues in the Database Engine. The session collects information that includes the following:

* The sql\_text and session\_id for any sessions that encounter an error that has a severity >=20.
* The sql\_text and session\_id for any sessions that encounter a memory-related error. The errors include 17803, 701, 802, 8645, 8651, 8657 and 8902.
* A record of any non-yielding scheduler problems. (These appear in the SQL Server error log as error 17883.)
* **Any deadlocks that are detected.**
* The callstack, sql\_text, and session\_id for any sessions that have waited on latches (or other interesting resources) for > 15 seconds.
* The callstack, sql\_text, and session\_id for any sessions that have waited on locks for > 30 seconds.
* The callstack, sql\_text, and session\_id for any sessions that have waited for a long time for preemptive waits. A preemptive wait is where SQL Server is waiting for external API calls.
* The callstack and session\_id for CLR allocation and virtual allocation failures.
* The ring\_buffer events for the memory broker, scheduler monitor, memory node OOM, security, and connectivity.
* System component results from sp\_server\_diagnostics.
* Instance health collected by scheduler\_monitor\_system\_health\_ring\_buffer\_recorded.
* CLR Allocation failures.
* Connectivity errors using connectivity\_ring\_buffer\_recorded.
* Security errors using security\_error\_ring\_buffer\_recorded.

The following trace flags should be activated by default in every instance of Microsoft SQL Server prior to SQL 2016 for performance reasons[[3]](#footnote-3):

|  |  |
| --- | --- |
| Trace flag | Description |
| 1117 | When a file in the file group meets the auto grow threshold, all files in the file group grow.  **Note**: Starting with SQL Server 2016 this behavior is controlled by the AUTOGROW\_SINGLE\_FILE and AUTOGROW\_ALL\_FILES option of ALTER DATABASE, and trace flag 1117 has no affect.  **Scope**: global only |
| 1118 | Removes most single page allocations on the server, reducing contention on the SGAM page. When a new object is created, by default, the first eight pages are allocated from different extents (mixed extents). Afterwards, when more pages are needed, those are allocated from that same extent (uniform extent). The SGAM page is used to track these mixed extents, so can quickly become a bottleneck when numerous mixed page allocations are occurring. This trace flag allocates all eight pages from the same extent when creating new objects, minimizing the need to scan the SGAM page. For more information, see this Microsoft Support article.  **Note:** Starting with SQL Server 2016 this behavior is controlled by the SET MIXED\_PAGE\_ALLOCATION option of ALTER DATABASE, and trace flag 1118 has no affect.  **Scope:** global only[[4]](#footnote-4). |
| 2371 | Changes the fixed auto update statistics threshold to dynamic auto update statistics threshold. For more information, see this Microsoft Support article.  **Note:** Starting with SQL Server 2016 this behavior is controlled by the engine and trace flag 2371 has no effect.  **Scope:** global only[[5]](#footnote-5). |

## Virtual Volumes

Microsoft SQL Server uses 4 drives for the storages of system databases and user databases.

|  |  |
| --- | --- |
| Drive | Usage |
| C: | OS-System |
| D: | System databases (master, model, msdb) |
| E: | Database files for user databases |
| F: | Log files for user databases |
| H: | Database and log files from system database TEMPDB |

Read and Write latency of some drives are suffering from bad performance:

| Drive | Read Latency | Write Latency | Overall Latency | Avg Bytes/Read | Avg Bytes/Write | Avg Bytes/Transfer |
| --- | --- | --- | --- | --- | --- | --- |
| D: | 31 | 20 | 22 | 131733 | 5950 | 37355 |
| E: | 61 | 10 | 51 | 806719 | 111642 | 673303 |
| F: | 5 | 4 | 4 | 671948 | 58215 | 66499 |
| H: | 3 | 8 | 6 | 61826 | 64604 | 63569 |

The read latency for drive E: is – based on the values – to slow. A very good performance counter for drive latency is <= 5 (very good), >5 and <= 10 (good), >10 and <=20 (average), >20 (below average).

Further investigation of the storage latency are required!

The size and physical placement of the tempdb database can affect the performance of a system. For example, if the size that is defined for tempdb is too small, part of the system-processing load may be taken up with auto growing tempdb to the size required to support the workload every time you restart the instance of SQL Server. This overhead can be avoid by increasing the sizes of the tempdb data and log file. The system database tempdb is used for several tasks inside the Database Storage Engine:

* Read Committed Snapshot Isolation[[6]](#footnote-6)
* HASH Join Operations[[7]](#footnote-7)
* SORT Operations[[8]](#footnote-8)
* Temporary Tables
* Table variables
* …

## Tempdb Size and Placement Recommendations

To achieve optimal tempdb performance, the following recommendation should take place for tempdb:

* Allow for tempdb files to automatically grow as required. This allows for the file to grow until the disk is full.

Note

If the production environment cannot tolerate the potential for application time-outs that may occur during auto grow operations, preallocate space to allow for the expected workload.

* **Set the file growth increment to a reasonable size to avoid the tempdb database files from growing by too small a value**. If the file growth is too small, compared to the amount of data that is being written to tempdb, tempdb may have to constantly expand. This will affect performance.
* Preallocation of space for all tempdb files by setting the file size to a value large enough to accommodate the typical workload in the environment. This prevents tempdb from expanding too frequently, which can affect performance. Tempdb database should be set to auto grow, but this should be used to increase disk space for unplanned exceptions.
* Create as many files as needed to maximize disk bandwidth. Using multiple files reduces tempdb storage contention and yields significantly better scalability. As a general guideline, one data file for each CPU on the server up to max 8 files (accounting for any affinity mask settings) should be created
* **Each data file has to be the same size; this allows for optimal proportional-fill performance.**
* Tempdb database has to be on a fast I/O subsystem.
* Tempdb database files should be stored on disks that differ from those that are used by user databases.

## Memory / RAM

The virtual environment is configured with 64 GB of memory. The available memory for Microsoft SQL Server has been limited to 60 GB.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPU | RAM (physically) | RAM (granted) | RAM (committed) | max\_workers\_count |
| 16 | 64.00 | 58.54 | 58.59 | 704 |

Based on the PLE (Page Life Expectancy[[9]](#footnote-9)) the memory seems sufficient. A big problem is the distribution of the cores on multiple NUMA Nodes (see page 9).

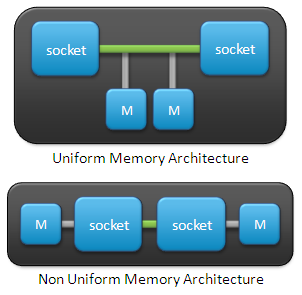
| object\_name | NUMA | PLE | Measure\_Date\_Time |
| --- | --- | --- | --- |
| SQLServer:Buffer Node | 0 | 48418 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 1 | 48419 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 2 | 48419 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 3 | 48419 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 4 | 48419 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 5 | 48416 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 6 | 48418 | 01.11.2017 15:07 |
| SQLServer:Buffer Node | 7 | 48418 | 01.11.2017 15:07 |

The analysis shows an average PLE of ~800 minutes per NUMA Node.

## NUMA-architecture

A weird configuration of NUMA configuration was found on the system. 16 vCores are distributed over 8 (!) NUMA nodes. Each NUMA node addresses 64 GB/8 = 16 GB of RAM.

Microsoft SQL Server is non-uniform memory access (NUMA) aware, and performs well on NUMA hardware without special configuration. As clock speed and the number of processors increase, it becomes increasingly difficult to reduce the memory latency required to use this additional processing power. NUMA architecture provides a scalable solution to address the memory. SQL Server has been designed to take advantage of NUMA-based computers without requiring any application changes.



Picture 1: NUMA

The trend in hardware has been towards more than one system bus, each serving a small set of processors. Each group of processors **has its own memory** and possibly its own I/O channels. However, each CPU can access memory associated with the other groups in a coherent way. Each group is called a NUMA node. The number of CPUs within a NUMA node depends on the hardware vendor. **It is faster to access local memory than the memory associated with other NUMA nodes.** This is the reason for the name, non-uniform memory access architecture.

On NUMA hardware, some regions of memory are on physically different buses from other regions. Because NUMA uses local and foreign memory, **it will take longer to access some regions of memory than others**. Local memory and foreign memory are typically used about a currently running thread. Local memory is the memory that is on the same node as the CPU currently running the thread. Any memory that does not belong to the node on which the thread is currently running is foreign. Foreign memory is also known as remote memory. The ratio of the cost to access foreign memory over that for local memory is called the NUMA ratio. If the NUMA ratio is 1, it is symmetric multiprocessing (SMP). **The greater the ratio, the more it costs to access the memory of other nodes**.

The main benefit of NUMA is scalability. The NUMA architecture was designed to surpass the scalability limits of the SMP architecture. With SMP, all memory access is posted to the same shared memory bus. **This works fine for a relatively small number of CPUs, but not when you have dozens, even hundreds, of CPUs competing for access to the shared memory bus.** NUMA alleviates these bottlenecks by limiting the number of CPUs on any one memory bus and connecting the various nodes by means of a high speed interconnection.

**Having too many NUMA nodes can lead to some really strange CPU behavior. The CPU workload of the machine cannot evenly balanced across the CPUs. If – in the current case – there will be 16 cores on the VM, configured with 8 NUMA nodes, it might be that two cores (might be four, or six it depends on a lot of factors) running very hot compared to the other cores on the server.**

## Microsoft SQL Server

## Configuration settings of Microsoft SQL Server

| Name | Description | value\_in\_use |
| --- | --- | --- |
| recovery interval (min) | Maximum recovery interval in minutes | 0 |
| locks | Number of locks for all users | 0 |
| fill factor (%) | Default fill factor percentage | 0 |
| cross db ownership chaining | Allow cross db ownership chaining | 0 |
| max worker threads | Maximum worker threads | 0 |
| cost threshold for parallelism | **cost threshold for parallelism** | **5** |
| max degree of parallelism | maximum degree of parallelism | 0 |
| min server memory (MB) | Minimum size of server memory (MB) | 16 |
| max server memory (MB) | Maximum size of server memory (MB) | 60000 |
| clr enabled | CLR user code execution enabled in the server | 1 |
| optimize for ad hoc workloads | **When this option is set, plan cache size is further reduced for single-use adhoc OLTP workload.** | **1** |
| Database Mail XPs | Enable or disable Database Mail XPs | 1 |
| xp\_cmdshell | **Enable or disable command shell** | **1** |

Some of the above settings have deep impacts to performance problems and can lead to slowly running systems

* Cost Threshold for Parallelism
* Max Degree of Parallelism
* Max Server Memory
* xp\_cmdshell

### Cost Threshold for Parallelism

The option “cost threshold for parallelism” specifies the threshold at which Microsoft SQL Server creates and runs parallel plans for queries. SQL Server creates and runs a parallel plan for a query only when the estimated cost to run a serial plan for the same query is higher than the value set in cost threshold for parallelism. The cost refers to an estimated elapsed time in seconds required to run the serial plan on a specific hardware configuration. Only set cost threshold for parallelism on symmetric multiprocessors.

Longer queries usually benefit from parallel plans; the performance advantage negates the additional time required to initialize, synchronize, and terminate parallel plans. The cost threshold for parallelism option is actively used when a mix of short and longer queries is run. The short queries run serial plans, whereas the longer queries use parallel plans. The value of cost threshold for parallelism determines which queries are considered short, and they should therefore be run using serial plans.

In certain cases, a parallel plan may be chosen even though the query's cost plan is less than the current cost threshold for parallelism value. This can happen because the decision to use a parallel or serial plan is based on a cost estimate provided before the full optimization is complete.

The cost threshold for parallelism option can be set to any value from 0 through 32767. **The default value is 5. This value is from 1997 (!) and is based on hardware from 1997.**

### Max Degree of Parallelism

The Microsoft SQL Server max degree of parallelism (MAXDOP) configuration option controls the number of processors that are used for the execution of a query in a parallel plan. **This option determines the computing and thread resources that are used for the query plan operators that perform the work in parallel**. Depending on whether SQL Server is set up on a symmetric multiprocessing (SMP) computer, a non-uniform memory access (NUMA) computer, or hyper threading-enabled processors, the option for the max degree of parallelism must be configured appropriately.

### Min/Max Server Memory

The two server memory options, “**min server memory**” and “**max server memory**” are used to reconfigure the amount of memory (in megabytes) that is managed by the SQL Server Memory Manager for a SQL Server process used by an instance of SQL Server.

The default setting for min server memory is 0, and the default setting for max server memory is 2147483647 MB. By default, SQL Server can change its memory requirements dynamically based on available system resources. When SQL Server is using memory dynamically, it queries the system periodically to determine the amount of free memory. Maintaining this free memory prevents the operating system (OS) from paging. If less memory is free, SQL Server releases memory to the OS. If more memory is free, SQL Server may allocate more memory. SQL Server adds memory only when its workload requires more memory; a server at rest does not increase the size of its virtual address space.

The distribution of system and user databases is described as follows[[10]](#footnote-10):

|  |  |  |
| --- | --- | --- |
| Buffer Pool Rank | database\_name | Cached Size |
| 1 | <DB\_1> | 13804,36 |
| 2 | <DB\_2> | 5969,54 |
| 3 | <DB\_3> | 918,47 |
| 4 | <DB\_4> | 620,86 |

The distribution picture shows a dominant consume of Memory by the databases <DB\_1> and <DB\_2>.

Note

The values are NOT representative for the production system because the UAT system is not heavily used.

### xp\_cmdshell

xp\_cmdshell is a security critical option in Microsoft SQL Server which spawns a Windows command shell and passes in a string for execution. Any output is returned as rows of text.

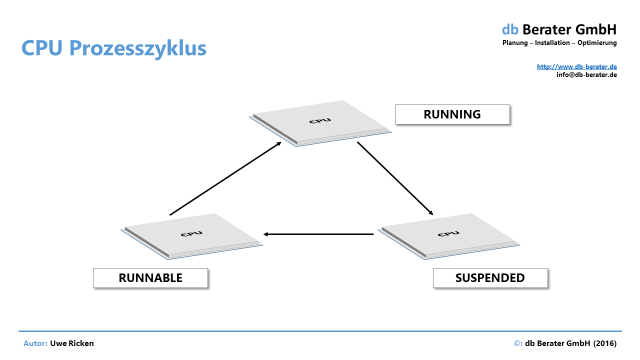
Note

* The Windows process spawned by xp\_cmdshell has the **same security rights as the SQL Server service account.**
* xp\_cmdshell operates synchronously. Control is not returned to the caller until the command-shell command is completed.

### Wait Stats

When a user application submits to SQL Server a request for data, the biggest element of SQL Server’s total response time would, ideally, be the CPU processing time. In other words, the time it takes the CPU to pull together the required data and send it back. However, in a busy database system, with hundreds or thousands of user requests competing for the limited resources of the database server, there will be times when a request is waiting to proceed, rather than actively processing. For example, Session A’s request may be waiting for Session B to release a lock on a resource to which it needs access.

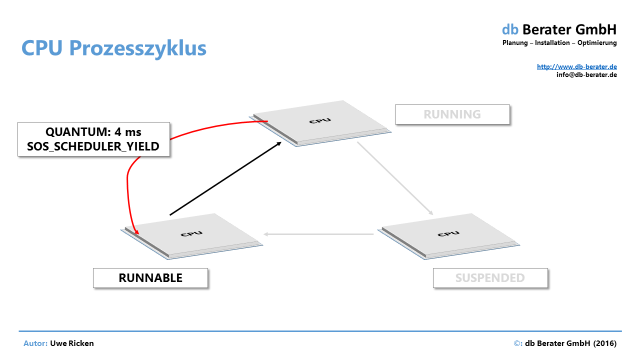
Every time a request is forced to wait, SQL Server records the length of the wait, and the cause of the wait, a.k.a. the wait type, which generally indicates the resource on which the request was waiting. These are the wait statistics.



Picture 2: Working cycle in Microsoft SQL Server

If a query is running actively on the CPU, the query is in the **RUNNING** state. Being in the running state is always the primary goal. When SQL Server takes a query off the CPU, then the query is moved into the **SUSPENDED** state. The query waits as long as needed in the SUSPENDED state, until the requested resource is available

When the requested resource is available, then SQL Server moves the query into the **RUNNABLE** state. The RUNNABLE state means that a query is ready for continuing its execution, but it needs one additional essential thing: a CPU to run on. When there is currently no CPU available (because other queries are currently in the RUNNING state), then the query has to spend some time in the RUNNABLE state. Finally when a CPU becomes available, the query moves into the RUNNING state, and everything happens over and over again.



Picture 3: Logging of a SOS\_SCHEDULER\_YIELD wait

| Wait Type | Wait (Sec) | Resource (Sec) | Signal (Sec) | WaitCount | % |
| --- | --- | --- | --- | --- | --- |
| CXPACKET[[11]](#footnote-11) | 253638,4 | 228542,8 | 25095,63 | 190134342 | 59,20 |
| OLEDB[[12]](#footnote-12) | 133336,8 | 133336,8 | 0,00 | 457884592 | 31,12 |
| LATCH\_EX[[13]](#footnote-13) | 14644,38 | 13650,18 | 994,20 | 30604267 | 3,42 |
| BACKUPBUFFER[[14]](#footnote-14) | 6375,05 | 6139,86 | 235,19 | 886630 | 1,49 |
| CMEMTHREAD | 5627,29 | 631,69 | 4995,60 | 183634695 | 1,31 |

The result demonstrates a heavy use of parallel running queries (CXPACKET und LATCH\_EX).

| Wait Type | Avg Wait\_S | Suspended (s) | Runnable (s) |
| --- | --- | --- | --- |
| CXPACKET | 0,0013 | 0,0012 | 0,0001 |
| OLEDB | 0,0003 | 0,0003 | 0,0000 |
| LATCH\_EX | 0,0005 | 0,0004 | 0,0000 |
| BACKUPBUFFER | 0,0072 | 0,0069 | 0,0003 |
| CMEMTHREAD | 0,0000 | 0,0000 | 0,0000 |

#### CXPACKET

This is a common wait that occurs when a parallel query is executing. During parallel processing, SQL Server breaks down queries into separate parts that can be processed simultaneously using multiple threads. Each thread **works on a separate piece of data** so that queries can be processed more quickly and efficiently. If work is not split equally between threads, those with fewer rows to process must wait until long-running threads complete.

CXPACKET waits can occur for different reasons in various environments, and you’ll need to find the root cause before deciding whether or not a problem exists. In an OLTP (Online Transaction Processing) with small transactions and short queries, excessive CXPACKET waits may affect throughput of other OLTP traffic.

#### LATCH\_EX

When an operation is waiting for exclusive write access to modify a structure in memory it records a LATCH\_EX wait. The structure is not related to buffers or transactions, but to some other process within SQL Server.

Latches are lightweight, performance-optimized mechanisms for data consistency within SQL Server. LATCH\_\* waits protect access to internal memory structures outside the buffer pool pages (non-buffer latches), and their use is determined by the SQL Server engine. The engine uses latch compatibility modes to manage concurrent attempts by **multiple threads** to acquire latches on the same structures.

|  |  |  |  |
| --- | --- | --- | --- |
| CPU Rank | Database Name | CPU\_Time\_Ms | CPU (%) |
| 1 | <DB\_1> | 1409,51 | 41,83 |
| 2 | <DB\_2> | 839,79 | 24,92 |
| 3 | <DB\_3> | 570,95 | 16,94 |
| 4 | <DB\_4> | 541,66 | 16,07 |
| 5 | <DB\_5> | 3,38 | 0,10 |

## Databases

### TEMPDB

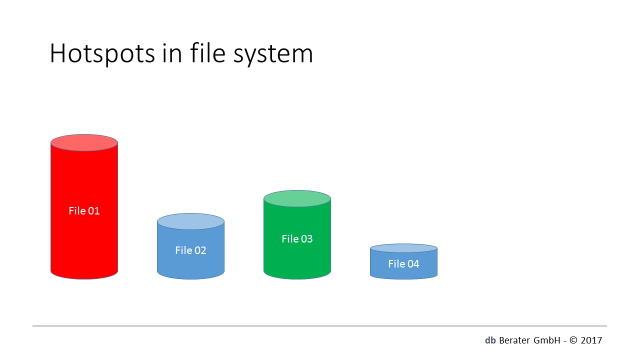
The size and physical placement of the tempdb database can affect the performance of a system. For example, if the size that is defined for tempdb is too small, part of the system-processing load may be taken up with auto growing tempdb to the size required to support the workload every time you restart the instance of SQL Server. To avoid this overhead the size of database files should be increased to an appropriate value for data and log file.

* If the production environment cannot tolerate the potential for application time-outs that may occur during auto grow operations, space need to be preallocate to allow for the expected workload.
* The file growth increment need to be set to a reasonable size to avoid the tempdb database files from growing by too small a value. If the file growth is too small, compared to the amount of data that is being written to tempdb, tempdb may have to constantly expand. This will affect performance.
* Space for all tempdb files should be set to a value large enough to accommodate the typical workload in the environment. This prevents tempdb from expanding too frequently, which can affect performance.
* The tempdb database should be set to auto grow, but this should be used to increase disk space for **unplanned** exceptions.
* As many files as needed need to be created to maximize disk bandwidth. Using multiple files reduces tempdb storage contention and yields significantly better scalability. The maximum files should be 8 for a system with more than 8 Cores. If the system has less or equal to 8 cores there should be one data file for each core!
* **Make each data file the same size; this allows for optimal proportional-fill performance**
* Tempdb database should be on a fast I/O subsystem.
* Tempdb database should be on disks that differ from those that are used by user databases.

TEMPDB is configured as follows:

| Name | physical\_name | type\_desc | size\_mb | growth\_mb |
| --- | --- | --- | --- | --- |
| tempdev1 | G:\...\tempdb.mdf | ROWS | 583,88 | 100,00 |
| tempdev2 | G:\...\tempdb2.mbf | ROWS | 479,81 | 100,00 |
| tempdev3 | G:\...\tempdb3.mbf | ROWS | 695,63 | 100,00 |
| tempdev4 | G:\...\tempdb4.mbf | ROWS | 100,00 | 100,00 |
| tempdev5 | G:\...\tempdb5.mbf | ROWS | 100,00 | 100,00 |
| tempdev6 | G:\...\tempdb6.mbf | ROWS | 100,00 | 100,00 |
| tempdev7 | G:\...\tempdb7.mbf | ROWS | 100,00 | 100,00 |
| tempdev8 | G:\...\tempdb8.mbf | ROWS | 100,00 | 100,00 |
| templog | H:\...\templog,ldf | LOG | 14,00 | 10,00 |

Tempdb is well configured based on the number of database files but is misconfigured in the size. All database Files (tempdev1 – to tempdev3) have different sizes which will lead to file access hotspots!



Picture 4: different size of database files

The example above shows a database with different file sizes. Because File 01 can save most of the data the data will be distributed uneven over all database files. File 04 is the smallest file and will not so often been used for storing data. If many writing processes are active in the system than all processes will try to write into File 01 and the database is suffering from write latencies.

### User database(s)

Note

This document will only cover user databases which are mandatory for the survey!

Configuration values

| database\_name | owner\_name | comp\_level | auto\_close | auto\_shrink | SI[[15]](#footnote-15) | RCSI[[16]](#footnote-16) |
| --- | --- | --- | --- | --- | --- | --- |
| <DB\_1> | sa | 90 | No | No | ON | Yes |
| <DB\_2> | sa | 90 | No | No | OFF | No |
| <DB\_3> | sa | 100 | No | No | OFF | No |
| <DB\_4> | sa | 90 | No | No | ON | Yes |
| <DB\_5> | sa | 90 | No | No | ON | Yes |
| <DB\_6> | sa | 100 | No | No | OFF | No |

#### Owner of databases

All the user databases have the local [sa] account as an owner.

#### Compatibility level

None of the databases is running with the compatibility level of Microsoft SQL Server 2012. The marked user databases are running with the compatibility level of Microsoft SQL Server 2005!

#### ALLOW\_SNAPSHOT\_ISOLATION

Snapshot isolation uses an optimistic concurrency model. If a snapshot transaction attempts to commit modifications to data that has changed since the transaction began, the transaction will roll back and an error will be raised. Snapshot isolation provides transaction-level read consistency. A data snapshot is taken when the snapshot transaction starts, and remains consistent for the duration of the transaction. Use snapshot isolation when:

* Optimistic concurrency control is desired.
* Probability is low that a transaction would have to be rolled back because of an update conflict.
* An application needs to generate reports based on long-running, multi-statement queries that must have point-in-time consistency. Snapshot isolation provides the benefit of repeatable reads (see Concurrency Effects) without using shared locks. Database snapshot can provide similar functionality but must be implemented manually. Snapshot isolation automatically provides the latest information in the database for each snapshot isolation transaction.

The option ALLOW\_SNAPSHOT\_ISOLATION is a MANUAL row versioning system which requires a dedicated switch by using “SET TRANSACTION LEVEL SNAPSHO”. If the application is not using it, it can be deactivated!

#### READ\_COMMITTED\_SNAPSHOT\_ISOLATION

The READ\_COMMITTED\_SNAPSHOT database option determines the behavior of the default READ COMMITTED isolation level when snapshot isolation is enabled in a database. If not explicitly specified, READ COMMITTED is applied to all implicit transactions. When READ\_COMMITTED\_SNAPSHOT database option is set to ON, the database engine uses row versioning and snapshot isolation **as the default**, instead of using locks to protect the data.

Once snapshot isolation is enabled, updated row versions for each transaction are maintained in **tempdb**. A unique transaction sequence number identifies each transaction, and these unique numbers are recorded for each row version. The transaction works with the most recent row versions having a sequence number before the sequence number of the transaction. Newer row versions created after the transaction has begun are ignored by the transaction.

# Recommendation

## Activation of default Trace flags

The following trace flags should be implemented as a standard to the UAT and PROD environment:

| Trace flag | Description |
| --- | --- |
| 1117 | When a file in the file group meets the auto grow threshold, all files in the file group grow.  <https://msdn.microsoft.com/en-us/library/ms188396.aspx> |
| 1118 | Removes most single page allocations on the server, reducing contention on the SGAM page.  <https://msdn.microsoft.com/en-us/library/ms188396.aspx> |
| 2371 | Changes the fixed auto update statistics threshold to dynamic auto update statistics threshold.  <https://support.microsoft.com/en-us/kb/2754171> |

## Configuration of TEMPDB

All files of the system database TEMPDB should have the same initial size to avoid a „hotspot contention“. All database files should be resized to an initial size of 512 MB to avoid auto growth.

## Cost Threshold for Parallelism

The values for the cost threshold for parallelism are outdated and from the early 90th of the last century. It is strongly recommended to change the value for a typical OLTP workload to a value of 25. For a DWH workload it should be set to 50!

EXEC sp\_configure N'show advanced options', 1;

RECONFIGURE WITH OVERRIDE;

GO

EXEC sp\_configure N'cost threshold for parallelism', 25;

RECONFIGURE WITH OVERRIDE;

GO

## NUMA Nodes

The system is wrong configured with the given setup of NUMA nodes. Microsoft SQL Server is working with 8 (!) NUMA nodes and 64 GB of RAM. The NUMA should be configured to 2 NUMA with 8 vCores for each NUMA node

## Max Degree of Parallelism

The MAXDOP (max degree of parallelism) slows down the system because every query which runs in parallel consumes all 16 vCores. When NUMA is configured correct the MAXDOP should be set to 4 vCores for each parallel query. With this setting Microsoft SQL Server can handle 4 queries at the same time when they all run in parallel!

EXEC sp\_configure N'show advanced options', 1;

RECONFIGURE WITH OVERRIDE;

GO

EXEC sp\_configure N'max degree of parallelism', 4;

RECONFIGURE WITH OVERRIDE;

GO

# Conclusion

The system is suffering from wrong configuration which is not best practice. Before running more surveys against the application, it is recommended to setup Microsoft SQL Server under “Best Practice”. When the UAT environment is correctly configured than we can run more surveys against the application itself.

65936 Frankfurt am Main, Germany – 11/07/2017

<AUTHOR>

1. <http://www.db-berater.de> [↑](#footnote-ref-1)
2. <https://www.microsoft.com/en-us/download/details.aspx?id=56040> [↑](#footnote-ref-2)
3. <https://docs.microsoft.com/en-us/sql/t-sql/database-console-commands/dbcc-traceon-trace-flags-transact-sql> [↑](#footnote-ref-3)
4. <https://technet.microsoft.com/en-us/library/ms190969.aspx> [↑](#footnote-ref-4)
5. <https://support.microsoft.com/en-us/help/2754171/controlling-autostat-auto-update-statistics-behavior-in-sql-server> [↑](#footnote-ref-5)
6. <https://technet.microsoft.com/de-de/library/ms189050.aspx> [↑](#footnote-ref-6)
7. <https://technet.microsoft.com/de-de/library/ms189313.aspx> [↑](#footnote-ref-7)
8. <https://technet.microsoft.com/de-de/library/ms191158.aspx> [↑](#footnote-ref-8)
9. <https://blogs.msdn.microsoft.com/psssql/2015/05/14/sql-server-page-life-expectancy-ple/> [↑](#footnote-ref-9)
10. Databases with a rate of less than 1% are not listed! [↑](#footnote-ref-10)
11. <http://documentation.red-gate.com/display/SM4/CXPACKET> [↑](#footnote-ref-11)
12. <http://documentation.red-gate.com/display/SM4/OLEDB> [↑](#footnote-ref-12)
13. <http://documentation.red-gate.com/display/SM4/LATCH_EX> [↑](#footnote-ref-13)
14. <http://documentation.red-gate.com/display/SM4/BACKUPBUFFER> [↑](#footnote-ref-14)
15. <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/snapshot-isolation-in-sql-server> [↑](#footnote-ref-15)
16. <https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/snapshot-isolation-in-sql-server> [↑](#footnote-ref-16)